UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/782,715	02/19/2004	Kevin Zatloukal	BEAS-01396US1 SRM/DTX	5577

23910 7590 09/15/2005

FLIESLER MEYER, LLP
FOUR EMBARCADERO CENTER
SUITE 400
SAN FRANCISCO, CA 94111

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 09/15/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/782,715	ZATLOUKAL, KEVIN	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-56* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-56* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *01 July 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *1/3,2/28,4/7,5/20*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated June 27, 2005.

2. Per Applicants' request, claims 1, 3, 7, 8, 9, 10, 11, 19, 21, 25, 26, 37, 39, 43, 44, 53, 55, and 56 have been amended.

3. Claims 1-56 remain pending.

Response to Amendment

4. Applicants' amendment dated 06/27/2005, responding to the 05/20/2005 Office action provided in the objection of drawings. The examiner has reviewed the updated drawings, Figures 1 and 2, respectfully. The set of formal drawings filed concurrently with the above-mentioned amendment is accepted by the Examiner

5. Applicant's response for 35 USC § 112 rejection of 05/20/2005 Office action (1st paragraph on page 18 of REMARKS) is reviewed and accepted by the examiner, the 35 USC § 112 rejection to claims 13, 31, and 49 is withdrawn.

6. Applicants' amendment dated 06/27/2005, responding to the 05/20/2004 Office action provided in the rejection of claims. The examiner has reviewed the amended claims, and noted that new matter has been introduced into the disclosure. See 35 U.S.C. 103 rejections herein below.

Response to Arguments

7. Applicant's arguments with respect to claims 1-62 have been considered but are moot in view of the new ground(s) of rejection necessitated by Applicant's amendments to the claims. For the Applicants' convenience, the 35 USC § 103 rejections are listed as following, with the amendments requested by the applicants.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1, 3-19, 21-37, 39-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6, 799,718 by Ken Chan et al., (hereinafter "Chan"), in view of U.S. Patent No. 6,836,883 by Abrams et al. (hereinafter "Abrams").

CLAIM	Chan / Abrams
1. A system operable to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising: (a) a compiler framework operable to perform a programming language independent portion of the compilation process on the computer program; (b) a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process on one of the one or more computer programming languages in the computer program and/or interact with another language module in the plurality of language modules; and	See Chan's Abstract, "Development assistance for a program comprising **code in more than one Language**", and column 2, lines 24-26, "Complete development assistance therefore requires not only solving the general problem of **multiple languages in the same file**, but any inter-language issues" and column 2, lines 52-53, "the present invention is directed to providing development assistance for multiple languages in an **Integrated Development Environment (IDE)**". For items and b, see Chan's Figure 3, the process to perform a programming language independent portion of the compilation (*process for each independent language, L1, L2, or L3*); and Figure 4 for each language, performs language-dependent portion of process. Chan teaches all aspects of claim 1, but does not mention the 'interact with another language module' specifically. However, Abrams teaches it in an analogous art. In Abrams, column 2, lines 41-54, "If the source language has no 'import' functionality, the **source syntax should be extended** to be able to achieve this

import functionality. Then the existing front end compiler system should be modified so that it can read at least the metadata portion of the intermediate file and appropriately handle the metadata. That is, the front end should be able to parse the common language file and convert the type and method information in the metadata into the proper form for the particular symbol table. For each new class that is added to the symbol table, the common language file must be read for the methods that are supported by the new class. If it is desired to also make direct use of the executable instructions of the common language file in the output, the front end may also need to be modified to appropriately read and handle the executable instructions." Also see Abrams' Fig. 3, the 'Execution Environment' functions like the compiler framework, which allows the interactive access to a different program language module.
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chan disclosure of the multi-programming-language environment by utilizing the ability to create a common language library (*information interface*) that can be used by developers using a number of different programming languages (see Abrams' column 12, lines 66-67).

(c) a plurality of language interfaces, wherein each language interface is provided by one of the plurality of a language modules to interact with the compiler framework.

Each language has its own interface (L1, has its own scanner S1, and parser P1, so does L2, and L3).

3. The system of claim 1, wherein: the system can be tailored for Java.

For the feature of claim 1 see claim 1 rejection. For the Java environment see Chan, column 1, lines 18-20, "these languages include both programming

	languages such as Pascal and **Java**, and markup languages like HTML or XML" and more description about the Java environment in column 12, line 29 into column 13, line 13.
4. The system of claim 1, wherein: the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.	For the feature of claim 1 see claim 1 rejection. The completion engine may access other **classes and source code** (files, paths) in the current **project**, and any classes and source code in **libraries used by the project**.
5. The system of claim 4, further comprising a type cache operable to: store types defined in the set of files in the project; store dependencies between the types it stores; and allow types defined in one programming language to reference types defined in another programming language.	For the feature of claim 4 see claim 4 rejection. For the rest of claim 5 feature see Chan column 9, lines 17-25, "The code for each language is parsed separately. L2 code, which was converted and **stored in file** 530, is parsed by the appropriate parser P2 (540). The parser 540 produces structure analysis 550 and **error information** 552 which may be stored together or separately. At least **error location information** (*a list of errors*) 554 is maintained while parsing. L3 code is parsed by the appropriate parser P3 (542) and the structure and error information 556 and 558 is **stored along** with **mapping information** at 560. *(maintaining the dependencies)*" And Chan's Claim 24, "determining the context of the code further comprises the steps of a. **referencing definition-type information contained in any portion of the mixed-language file**; and b. determining the context of the code based on the referenced **definition-type information**".
6. The system of claim 4, wherein: the programming language-independent portion of the compilation process comprises at least one of the following phases:	See claim 5 rejection.

(a) managing the set of files in the project;
(c) persisting the set of paths files in the project;
(d) maintaining the set of dependencies in the project;
(e) acquiring configuration information files in the project; and
(f) maintaining a list of errors related to the project.

7. The system of claim 1, wherein:
the programming language-dependent portion of the compilation process comprises at least one of the following phases;
(a) performing scanning;
(b) performing parsing;
(c) performing name resolution;
(d) performing semantic checking; and
(e) performing code generation.

For the feature of claim 1 see claim 1 rejection. Chan's parser, scanner, and scanner generator cover the 'scanning', 'parsing', 'name resolution', 'semantic checking' and 'code generation' (typical compiler work) functions.

8. The system of claim 7, wherein:
a language interface of the plurality of language interfaces can be configured to present each language module in the form of a set of components, wherein each component is configured to perform one phase of the language-dependent portion of the compilation process.

For the feature of claim 7 see claim 7 rejection. In Chan's disclosure, the P1, P2, P3, S1, S2, S3, E1 (Completion Engine), E2, and E3 are a set of components, each components performing one phase of the compilation process.

9. The system of claim 1, wherein:
a language interface of the plurality of language interfaces can include functions for retrieving information about a particular file in the computer program.

For the feature of claim 1 see claim 1 rejection. In Chan's disclosure the P1, P2 and P3 are the parsers for different languages L1, L2, and L3; each of them is a language interface of the plurality of language interfaces that include functions for retrieving information about a particular file in a computer program.

10. The system of claim 1, wherein:
a language interface of the plurality of language interfaces can be donfigured to allow a first language module of the

For the feature of claim 1 see claim 1 rejection. For the rest of claim 10 feature see claim 4 and 5 rejections.

plurality of language modules to interact with a second language module of the plurality of language modules.	
11. The system of claim 10, wherein: the language interface is operable to accept nested languages, allowing the first language module to request the compilation of a specified portion of the computer program using the second language module.	For the feature of claim 10 see claim 10 rejection. For the rest of claim 11 feature see claim 4 and 5 rejections (the **definition-type information** files are sharable for different languages).
12. The system of claim 1, wherein: at least one language module of the plurality of language modules is for Java language.	For the feature of claim 1 see claim 1 rejection. For the rest of claim 12 feature see claim 3.
13. The system of claim 1, wherein: a second language module of the plurality of language modules extends a first language module of the plurality of language modules to provide a new language which is an extended version of a programming language compiled by the first language module.	For the feature of claim 1 see claim 1 rejection. For the rest of claim 11 feature see claim 4 and 5 rejections (the **definition-type information** files are sharable for different languages).
14. The system of claim 1, further comprising: a tool to speed the development of the plurality of language modules.	See Chan column 1, lines 13-18, "An Integrated Development Environment (IDE) is an application or set of program modules run from a single user interface which provides an environment for the development of application programs in particular computer languages. IDEs typically provide development assistance for the language or languages the IDE supports." – IDE is a tool that can speed the development of the plurality of language modules.
15. A system operable to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising:	See claim 1, claim 5, and claim 9 rejections.

(a) a compiler framework operable to perform a language-independent portion of the compilation process on the computer program;
(b) a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process on one of the one or more computer programming languages in the computer program and to provide language information about the computer program;
(c) an information interface operable to permit each client in a plurality of clients to interact with the compiler framework;
(d) a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks;
obtaining the language information produced by the plurality of language modules; and
requesting a service provided by the compiler framework.

16. The system of claim 15, wherein the plurality of clients comprise:
an integrated development environment (IDE) for developing a computer program wherein the IDE uses the language information from the compiler framework to provide language features based on that information; and

a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.

For the feature of claim 15 see claim 15 rejection. Chan's disclosure specifically teaches using IDE for a multi-programming language environment, see claim 1 rejection.

17. The system of claim 16, wherein:
the IDE may include a source code editor to edit files in the computer program.

For the feature of claim 16 see claim 16 rejection. IDE provide the editing feature, see Chan column 5, lines 26-33, "the features of development assistance

	provided in the **IDE** include: syntax highlighting, structure analysis, error reporting, completion assistance, and context-sensitive help. Referring to FIGS. 1 and 2, the **IDE is a graphical user interface** 10 on a Windows platform in which the subject file M is being developed. The code being developed in M is illustrated in a **text editor** window 11, in which syntax highlighting is visible."
18. The system of claim 15, wherein: the information interface is further operable to allow a client to inform the compiler network of file changes; and the compiler network is operable to recompile the changed files and other files depend on them.	For the feature of claim 15 see claim 15 rejection. For the rest of claim 18 feature see Chan column 11, lines 46-55, the mapping information provides the 'dependencies' that would inform the compiler network of file changes, and other files depend on them.
19. A method operable to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising: (a) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process on the computer program; (b) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process on one of the one or more computer programming languages in the computer program and/or interact with another language module in the plurality of language modules; and (c) providing a plurality of language interfaces, wherein each language interface is operable to allow one of the plurality of a language modules to interact with the compiler framework.	Same as claim 1 and claim 9 rejections.

21. The method of claim 19, further comprising: tailoring the compilation process for Java.	For the feature of claim 19 see claim 19 rejection. For 'Java' feature see claim 3 rejection.
22. The method of claim 19, wherein: the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.	For the feature of claim 19 see claim 19 rejection. For 'project' features see claim 4 rejection.
23. The method of claim 22, further comprising: (a) utilizing a type cache operable to; (b) store types defined in the set of files in the project; (c) store dependencies between the types it stores; and (d) allow types defined in one programming language to reference types defined in another programming language.	For the feature of claim 22 see claim 22 rejection. For the rest of the features see claim 5 rejection.
24. The method of claim 22, wherein: the programming language independent portion of the compilation process comprises at least one of the following phases; (a) managing the set of files in the project; (b) persisting the set of paths files in the project; (c) maintaining the set of dependencies in the project; (d) acquiring configuration information files in the project; and (e) maintaining a list of errors related to the project.	For the feature of claim 22 see claim 22 rejection. For the rest of the features see claim 6 rejection.
25. The method of claim 19, wherein: the programming language-dependent portion of the compilation process comprises at least one of the following phases; (a) performing scanning;	For the feature of claim 19 see claim 19 rejection. For the rest of the features of claim 25, see claim 7 rejection.

(b) performing parsing;
(c) performing name resolution;
(d) performing semantic checking; and
(e) performing code generation.

26. The method of claim 25, further comprising: presenting the each language module in the form of a set of components, wherein each component is configured to perform one phase of the language-dependent portion of the compilation process.	For the feature of claim 25 see claim 25 rejection. For the rest of the features of claim 26, see claim 8 rejection.
27. The method of claim 19, further comprising: retrieving information about a particular file in the computer program via a language interface of the plurality of language interfaces.	For the feature of claim 19 see claim 19 rejection. For rest of claim 27 feature see claim 9 rejection.
28. The method of claim 19, further comprising: allowing a first language module of the plurality of language modules to interact with a second language module of the plurality of language modules.	For the feature of claim 19 see claim 19 rejection. For rest of claim 28 feature see claim 10 rejection.
29. The method of claim 28, wherein: the first language module is operable to request the compilation of a specified portion of the computer program using the second language module.	For the feature of claim 28 see claim 28 rejection. For rest of claim 29 feature see claim 11 rejection.
30. The method of claim 19, wherein: at least one language module of the plurality of language modules is for Java language.	For the feature of claim 19 see claim 19 rejection. For rest of claim 30 feature see claim 3 rejection.
31. The method of claim 19, further comprising: extending a first language module of the plurality of language modules using a second language module of the plurality of language modules to provide a new	For the feature of claim 19 see claim 19 rejection. For rest of claim 31 feature see claim 13 rejection.

language which is an extended version of a programming language compiled by the first language module.

32. The method of claim 19, further comprising:
adopting tools to speed up the development of the plurality of language modules.

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 43, see claim 7 rejection.

33. A method operable to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising:
(a) utilizing a compiler framework operable to perform a language independent portion of the compilation process on the computer program;
(b) invoking a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process on one of the one or more computer programming languages in the computer program and to provide language information about the computer program;
(c) providing an information interface operable to permit each client in a plurality of clients to interact with the compiler framework;
(d) including a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks;
(i) obtaining the language information produced by the plurality of language modules; and
(ii) requesting a service provided by the compiler framework.

Same as claim 1 and claim 9 rejections.

34. The method of claim 33, wherein the plurality of clients comprise:

For the feature of claim 33 see claim 33 rejection. For the rest of the features of

(a) an integrated development environment (IDE) for developing a computer program wherein the IDE uses the language information from the compiler framework to provide language features based on that information; and (b) a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.	claim 34, see claim 16 rejection.
35. The method of claim 34, further comprising: including a source code editor in the IDE to edit files in the computer program.	For the feature of claim 34 see claim 34 rejection. For the rest of the features of claim 35, see claim 17 rejection.
36. The method of claim 33, further comprising: allowing a client to inform the compiler network of file changes; and recompiling the changed files and other files depend on them.	For the feature of claim 33 see claim 33 rejection. For the rest of the features of claim 36, see claim 3 rejection.
37. A machine readable medium having instructions stored thereon that when executed by a processor cause a System to: (a) perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising; (b) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process on the computer program; (c) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process on one of the one or more computer programming languages in the computer program and/or interact with	Chan's disclosure is a machine readable medium having instructions stored to compile a multi-language program. For the claim 37 features, see claim 1, claim 8, and claim 9 rejections.

another language module in the plurality of language modules; and
(d) providing a plurality of language interfaces, wherein each language interface is operable to allow one of the plurality of language modules to interact with the compiler framework.

39. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
tailor the compilation process for Java.

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 39, see claim 3 rejection.

40. The machine readable medium of claim 37, wherein:
the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 40, see claim 4 rejection.

41. The machine readable medium of claim 40, further comprising instructions that when executed cause the system to:
(a) utilize a type cache operable to;
(b) store types defined in the set of files in the project;
(c) store dependencies between the types it stores; and
(d) allow types defined in one programming language to reference types defined in another programming language.

For the feature of claim 40 see claim 40 rejection. For the rest of the features of claim 41, see claim 5 rejection.

42. The machine readable medium of claim 40, further comprising instructions that when executed cause the system to:
perform the programming language-independent portion of the compilation process in at least one of the following phases;
(a) managing the set of files in the project;
(b) persisting the set of paths files in the project;

For the feature of claim 40 see claim 40 rejection. For the rest of the features of claim 42, see claim 6 rejection.

(c) maintaining the set of dependencies in the project;
(d) acquiring configuration information files in the project; and
(e) maintaining a list of errors related to the project.

43. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
perform the programming language-dependent portion of the compilation process in at least one of the following phases;
(a) performing scanning;
(b) performing parsing;
(c) performing name resolution;
(d) performing semantic checking; and
(e) performing code generation.

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 43, see claim 7 rejection.

44. The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:
present each the language module in the form of a set of components, wherein each component is configured to perform one phase of the language-dependent portion of the compilation process.

For the feature of claim 43 see claim 43 rejection. For the rest of the features of claim 44, see claim 8 rejection.

45. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
retrieve information about a particular file in the computer program via a language interface of the plurality of language interfaces.

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 45, see claim 9 rejection.

46. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
allow a first language module of the plurality of language modules to interact with a second language module of the

For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 46, see claim 10 rejection.

plurality of language modules.

47. The machine readable medium of claim 37, wherein: the first language module is operable to request the compilation of a specified portion of the computer program using the second language module.	For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 47, see claim 11 rejection.
48. The machine readable medium of claim 37, wherein: at least one language module of the plurality of language modules is for Java language.	For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 48, see claim 3 rejection.
49. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: extend a first language module of the plurality of language modules using a second language module of the plurality of language modules to provide a new language which is an extended version of a programming language compiled by the first language module.	For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 49, see claim 13 rejection.
50. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: adopt tools to speed up the development of the plurality of language modules.	For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 50, see claim 14 rejection.
51. A machine readable medium having instructions stored thereon that when executed by a processor cause a system to: perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising: (a) utilize a compiler framework operable to perform a language-independent-portion of the compilation process on the computer program;	See claim 37 rejection.

(b) invoke a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process on one of the one or more computer programming languages in the computer program and to provide language information about the computer program;

(c) provide an information interface operable to permit each client in a plurality of clients to interact with the compiler framework;

(d) include a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks;

(i) obtaining the language information produced by the plurality of language modules; and

(ii) requesting a service provided by the compiler framework.

52. The machine readable medium of claim 51, wherein the plurality of clients comprise:

(a) an integrated development environment (IDE) for developing a computer program wherein the DE uses the language information from the compiler framework to provide language features based on that information; and

(b) a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.

For the feature of claim 51 see claim 51 rejection. For the rest of the features of claim 52, see claim 16 rejection.

53. The machine readable medium of claim 52, further comprising instructions that when executed cause the system to: include a source code editor in the IDE to edit files in the computer program.

For the feature of claim 52 see claim 52 rejection. For the rest of the features of claim 53, see claim 17 rejection.

54. The machine readable medium of

For the feature of claim 51 see claim 51

claim 51, further comprising instructions that when executed cause the system to: allow a client to inform the compiler network of file changes; and recompile the changed files and other files depend on them.	rejection. For the rest of the features of claim 54, see claim 18 rejection.
55. A system operable to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising: (a) means to utilize a compiler framework operable to perform a programming language-independent portion of the compilation process on the computer program; (b) means to invoke a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process on one of the one or more computer programming languages in the computer program and/or interact with another language module in the plurality of language modules; and (c) means to provide a plurality of language interfaces, wherein each language interface is operable to allow one of the plurality of language modules to interact with the compiler framework.	See claim 1, claim 8, and claim 9 rejections.
56. A computer data signal embodied in a transmission medium, comprising: (a) a code segment including instructions to perform a multi-programming-language compilation process on a computer program written in one or more computer programming languages, comprising; (i) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process on the computer program;	See claim 1, claim 8, and claim 9 rejections.

(ii) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process on one of the one or more computer programming languages in the computer program and/or interact with another language module in the plurality of language modules; and

(iii) providing a plurality of language interfaces, wherein each language interface is operable to allow one of the plurality of a language modules to interact with the compiler framework.

13. Claims 2, 20, and 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6, 799,718 by Ken Chan et al., (hereinafter "Chan"), in view of U.S. Patent No. 6,836,883 by Abrams et al. (hereinafter "Abrams"), and further in view of U.S. Patent No. 6,804,686 by Blake W. Stone et al. (hereinafter "Stone").

CLAIM	Chan / Abrams / Stone
2. The system of claim 1, wherein: a multi-threading service is used by the compiler framework and the plurality of language modules.	For the feature of claim 1 see claim 1 rejection. Chan and Abrams teach all aspects of claim 2 and 3, but does not mention the 'multi-threading' specifically. However, Stone teaches these features in an analogous art. In Stone, column 8, lines 12-24, "**Java** is a simple, object-oriented language which supports **multi-thread** processing and garbage collection. ...**Java programs** are 'compiled' into a binary format that can be executed on many different platforms without recompilation. A typical **Java** system comprises the following set of interrelated technologies: a language specification; a compiler for the **Java** language that produces bytecodes from an abstract, stack-oriented machine; a virtual machine (VM) program that interprets the bytecodes at runtime; a set of class libraries; a runtime environment that includes bytecode verification, **multi-**

	threading, and garbage collection;" It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chan and Abrams' disclosures of the multi-programming-language environment, able to access modules from a different programming language, by utilizing Java environment and multi-threading service taught by Stone, for the purpose of deploying application code efficiently to multiple platforms. (Stone, column 2, lines 7-8).
20. The method of claim 19, further comprising: utilizing a multi-threading service during the compilation process.	For the feature of claim 19 see claim 19 rejection. For 'multi-threading' feature see claim 2 rejection.
38. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: use a multi-threading service during the compilation process.	For the feature of claim 37 see claim 37 rejection. For 'multi-threading' feature see claim 2 rejection.

Conclusion

The following summarizes the status of the claims:

35 USC § 103 rejection: Claims 1-56

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any

extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow

Examiner

Art Unit 2192

September 02, 2005

cc

ANTONY NGUYEN-BA
PRIMARY EXAMINER